**Azure**

Install and use Hortonworks on Azure

<https://bigdata-etl.com/how-to-install-hortonworks-in-azure/>

<https://bigdata-etl.com/how-to-install-hortonworks-sandbox-with-data-platform-in-microsoft-azure-part-2/>

<https://www.youtube.com/watch?v=VefG02MA0Ik&t=519s>

**Connecting and loading data to HIVE:**

**hive jdbc driver:**

jdbc:hive2://sandbox-hdp.hortonworks.com:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2

CREATE TABLE CustomerOrders

(CustomerName VARCHAR(30),

OrdDate DATE,

Category CHAR(1),

ProductName VARCHAR(30),

Quantity INT,

Price DECIMAL(5,2)

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

tblproperties ("skip.header.line.count"="1")

;

LOAD DATA INPATH '/demo/data/Customer/Customer-Orders.csv'

OVERWRITE INTO TABLE customerorders;

**Azure account**

[K\_rasuri@outlook.com](mailto:K_rasuri@outlook.com)

Submit.27

**HortonWorks sandbox**

hdpuser

Generate.123

root

Admin123

**Ambari connection**

http://hwsandbox.eastus.cloudapp.azure.com:8888

maria\_dev

maria\_dev

**Connecting to Hortonworks VM from Azure CLI**

ssh [hdpuser@40.112.53.25](mailto:hdpuser@40.112.53.25)

ssh –p 2222 root@localhost

beeline

!connect jdbc:hive2://sandbox-hdp.hortonworks.com:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2;

**HDInsight Cluster:**

admin, Generate.123

sshuser, Generate.123

hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/sample\_08/sample\_08.csv

**val** textFile **=** spark.read.textFile("hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/sample\_08/sample\_08.csv

")

**Exercises:**

* PySpark commands from spark shell

<https://spark.apache.org/docs/2.4.0/quick-start.html>

* Build a program from Visual studio code. Deploy it to the cluster and run it
* Run an actual program on 6 node cluster
* Dynamically switch datasets in Azure Data Factory
* Slowly changing dimensions in ADF
* Parse Json, XML in databricks
* Working with Parquet, ORC and AVRO formats

Capturing events through Azure Event Hubs

<https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-capture-overview>

Integration run time azure data factory

<https://docs.microsoft.com/en-us/azure/data-factory/concepts-integration-runtime>

Access to azure storage

<https://docs.microsoft.com/en-us/azure/storage/common/storage-auth?toc=/azure/storage/blobs/toc.json>

Processing the events using event hubs

<https://www.youtube.com/watch?v=Dc3P27BsK3E>

Perform ETL in Snowflake using Stored Procedures, Azure Data Factory, and Azure Functions

<https://www.youtube.com/watch?v=ydMWEczHO2E>

HDInsight cluster types

<https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-hadoop-provision-linux-clusters>

Build spark application using IntelliJ, deploy and run it on HDInsight cluster

<https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-create-standalone-application>

<https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-intellij-tool-plugin>

Manage Cluster resources

<https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-resource-manager>

Monitor spark applications using history server

<https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-azure-spark-history-server>

<https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-log-management>

Selecting the right VM for your HDInsight cluster

<https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-selecting-vm-size>

Execute spark and Hive queries from Visual Studio Code

<https://docs.microsoft.com/en-us/azure/hdinsight/hdinsight-for-vscode>

Create HDInsight cluster from scripts

<https://docs.microsoft.com/en-us/azure/hdinsight/azure-cli-samples>

<https://github.com/Azure-Samples/hdinsight-dotnet-sdk-samples>

<https://docs.microsoft.com/en-us/python/api/overview/azure/hdinsight?view=azure-python>

Debugging failed Spark job in IntelliJ

<https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-intellij-tool-failure-debug>

<https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-intellij-tool-plugin-debug-jobs-remotely>

Improve spark jobs performance

<https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-improve-performance-iocache>

<https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-perf>

<https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-settings>

Spark performance and partition strategies

<https://medium.com/datalex/on-spark-performance-and-partitioning-strategies-72992bbbf150>

How Apache Spark works

<https://www.dezyre.com/article/apache-spark-architecture-explained-in-detail/338>

Determining the number of partitions in Apache Spark

<https://medium.com/dataseries/determining-number-of-partitions-in-apache-spark-part-i-e21a9ced6ad4>

PySpark Examples

<https://sparkbyexamples.com/pyspark-tutorial/>

<https://github.com/spark-examples/pyspark-examples/>

<https://github.com/apache/spark/tree/master/examples/src/main/python>

Data Modeling CosmosDB

<https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-model-partition-example>

<https://docs.microsoft.com/en-us/azure/cosmos-db/modeling-data#hybrid-data-models>

Difference between NO SQL and Relational databases

<https://docs.microsoft.com/en-us/azure/cosmos-db/relational-nosql>

Consistency Levels in CosmosDB

<https://docs.microsoft.com/en-us/azure/cosmos-db/consistency-levels>

Partitioning in CosmosDB

<https://docs.microsoft.com/en-us/azure/cosmos-db/partitioning-overview>

<https://docs.microsoft.com/en-us/azure/cosmos-db/how-to-model-partition-example>

<https://www.youtube.com/watch?v=KlXhee6R0rE>

<https://www.youtube.com/watch?v=9v6WbCOzPiM>

High Availability in CosmosDB

<https://docs.microsoft.com/en-us/azure/cosmos-db/high-availability>

Firewall rules for SQL Database

<https://docs.microsoft.com/en-us/azure/azure-sql/database/firewall-configure>

<https://docs.microsoft.com/en-us/learn/modules/secure-your-azure-sql-database/2-restrict-network-access>

Designing Distributed tables in Synapse

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-tables-distribute>

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/massively-parallel-processing-mpp-architecture>

Database Migration Assistant

<https://docs.microsoft.com/en-us/sql/dma/dma-overview?view=sql-server-ver15>

SQL Server Migration Assistant

<https://docs.microsoft.com/en-us/sql/ssma/sql-server-migration-assistant?view=sql-server-ver15>

Azure Database Migration Service

<https://docs.microsoft.com/en-us/azure/dms/dms-overview>

<https://www.youtube.com/watch?v=eiSjf5czO0g>

Migration using BACPAC

<https://docs.microsoft.com/en-us/azure/azure-sql/database/database-import?tabs=azure-powershell>

Using Stream Analytics with Azure Synapse

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-integrate-azure-stream-analytics>

SQL Server database security overview

<https://docs.microsoft.com/en-us/azure/azure-sql/database/security-overview>

Data Discovery and Classification in Azure SQL database

<https://docs.microsoft.com/en-us/azure/azure-sql/database/data-discovery-and-classification-overview>

Load data into Synapse using Polybase

<https://docs.microsoft.com/en-us/azure/synapse-analytics/sql-data-warehouse/sql-data-warehouse-load-from-azure-blob-storage-with-polybase>

Managing elastic pools in SQL Databases

<https://docs.microsoft.com/en-us/azure/azure-sql/database/elastic-pool-overview>

Incremental data load

<https://docs.microsoft.com/en-us/azure/data-factory/tutorial-incremental-copy-overview>

<https://docs.microsoft.com/en-us/azure/data-factory/tutorial-incremental-copy-powershell>

Iterate over a data collection using lookup and for each loop activities

<https://www.youtube.com/watch?v=ROq5mVrZPY0&t=1442s>

Move files with data factory

<https://www.data4v.com/move-files-with-azure-data-factory-end-to-end/>

Parameterize Azure Data Factory

<https://docs.microsoft.com/en-us/azure/data-factory/parameterize-linked-services>

<https://www.youtube.com/watch?v=pISBgwrdxPM>

ARM templates

<https://www.youtube.com/watch?v=3xbZJT3hTsQ>

<https://andyleonard.blog/2019/06/using-azure-resource-manager-templates-with-azure-data-factory/>

Azure Data Factory Deployment

<https://nikolaithomassen.com/2019/01/20/azure-devops-with-data-factory/>

Slowly changing dimension with ADF

<https://www.youtube.com/watch?v=ps12o93VAo0>

Loading data to SQL server from azure Databricks

<https://www.sqlshack.com/load-data-into-azure-sql-database-from-azure-databricks/>

HDInsights deployment from CLI

<https://github.com/Azure-Samples/hdinsight-sales-insights-etl>

Azure code samples on Git

<https://github.com/Azure-Samples>

Open datasets

<https://azure.microsoft.com/en-us/services/open-datasets/#overview>

<https://www.kaggle.com/datasets>

<https://www.transtats.bts.gov/Fields.asp>

<https://openflights.org/data.html>

<https://opendata.cityofnewyork.us/data/>

<https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>

What are SQL Server big data clusters?

<https://docs.microsoft.com/en-us/sql/big-data-cluster/big-data-cluster-overview?view=sql-server-ver15>

ETL with Databricks and Synapse

<https://docs.microsoft.com/en-us/azure/databricks/scenarios/databricks-extract-load-sql-data-warehouse?toc=%2Fazure%2Fstorage%2Fblobs%2Ftoc.json>

Creating HDFS user

<https://community.cloudera.com/t5/Support-Questions/Closed-How-to-create-user-in-HDFS/td-p/203448>

Adding columns to existing HIVE table

<http://dmtolpeko.com/2015/01/16/adding-columns-to-an-existing-table-in-hive/>

Working with HIVE external tables

<http://myitlearnings.com/creating-external-hive-table-and-import-data/>

Data Skew problem in Apache Spark

<https://blog.clairvoyantsoft.com/optimize-the-skew-in-spark-e523c6ee18ac>

Improve data lake storage performance

<https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-performance-tuning-guidance>

Data Factory copy activity performance features

<https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-performance-features>

Overview of Apache Kafka

<https://www.youtube.com/watch?v=Yc9AWb2hBJ0&list=PL0Uw5OUr1nQitAdwgIyGzvsRmTeWG4TBf>

**Spark Performance Optimization:**

* **Infer** **schema** scans the whole table to find out the data types of the columns. For small datasets it is ok, but for large data it’s better to avoid and provide our own schema while reading the data.
* Don’t use **collect** method on large datasets. Use **take()** method instead. Collect will bring all the data on all nodes to the driver, running the driver out of memory
* Use **reduce\_by\_key** when possible because it will aggregate the data in the partition first before joining with other partitions.
* Use **broadcast** of small tables whenever possible because join happens in the mapper stage and we will be able to avoid shuffling of data
* When there is a **data skew** problem it will hold up the whole job. In this case it is better to use bucketing and join based on the **partition** and **bucketing key**
* Increasing the number of nodes beyond the available keys count won’t help in any performance improvement
* **Filter** the data on the **source side** before loading the data into spark.
* Use the file formats like **parquet** and **ORC** because they provide better compression. Less data fast processing
* Configure the number of **executors, partitions size and parallelism** to achieve better performance. Degree of parallelism. How many tasks you want to execute in parallel. Default is set to 200 but it is configurable
* **Vectorization** will execute the data in batches of 1024 records instead of single row.
* **Indexing** provides better performance as well. Indexing will create a separate table which will refer the original table for actual data
* Zip files are **not splittable**. All unzipping process will be done on only one core and it will block the job until it completes. It’s better to unzip the files before processing them with spark
* **Too many partitions** will pose a problem too. There is an overhead for maintaining the metadata. We can use **coalesce** to reduce the partitions but it doesn’t guarantee similar data present in same partition, partitions may be uneven. It’s better to **repartition** by providing a key. More shuffling happens in repartition so it is comparatively slower than coalesce but it provides better partitioning.
* **Cache** the reusable datasets in the memory

spark.conf.get("spark.databricks.clusterUsageTags.clusterName")

spark.conf.get("spark.sql.shuffle.partitions")

spark.conf.get("spark.sql.files.maxPartitionBytes")

spark.conf.get("spark.sql.autoBroadcastJoinThreshold")

spark.conf.set("spark.executor.memory","4028")

spark.conf.set("spark.sql.shuffle.partitions", 20)